CHAPTER

14

ACTIVE DIRECTORY REPLICATION

After completing this chapter, you will be able to:

- Describe how the Windows 2000 domain controllers replicate data to each other
- Explain how Windows 2000 minimizes the amount of data that is replicated
- Explain how Windows 2000 prevents unnecessary data being copied across the network
- Describe what happens when simultaneous updates occur on a single object
- Describe the processes that Active Directory uses to prevent replication from swamping WAN links

If you are new to Active Directory, and if you have been paying close attention to the information in this book, you have probably already formulated lots of questions. In Chapter 2, you were introduced to the new terminology of Windows 2000 and to various components that have an effect on Active Directory. Now we are going to put some of that new terminology into practice, and at the same time we will answer some of the most pressing questions you might have regarding Active Directory.

Introduction to Active Directory Replication

Active Directory allows changes to occur at any domain controller, which is in contrast to Microsoft Windows NT 4 and earlier, where every change must occur at the Primary Domain Controller (PDC). In earlier versions, these changes are then replicated out to Backup Domain Controllers (BDCs) that contain read-only copies of domain data. It's fairly easy to understand the older model: Changes occur in one place and one place only, and they are then effectively copied everywhere else. In Windows 2000, changes can happen anywhere—so how is that information replicated (or copied) everywhere on the network? The answer is a complex mix of property versioning and in-memory tables that will be unraveled in this chapter.

Because a change can occur at any domain controller (DC), Active Directory has a problem. Let's say we make a change to a user's last name at a DC at the head office in Houston, Texas. The user actually works five blocks away and is authenticated by a local DC. What changes are sent from the DC in the head office to the local office? Does it make sense to send the entire copy of the directory? Maybe the DC should send all the information it has regarding the user. Microsoft has been careful to make sure that unnecessary data is not replicated on the network. As you will soon see, Microsoft's design goals are slightly different depending on whether the user's DC is on the same TCP/IP (IP) subnet as the DC in the head office.

When a change occurs at a DC in Windows 2000, the change causes the DC to let its partners know it has accepted a change. That triggers a replication process (we'll dig deep into this process later). This occurs for *every* change. So, let's take a scenario with three DCs. A change occurs at one, and the DC sends a message saying, "I've got a change" to a partner DC. That DC makes the change. Then *both* DCs let the third DC know that they have accepted changes. Is the change sent from both controllers? If it is, does the third controller write that change twice? What would prevent it from writing the change twice? We'll be taking a close look at the resolution process and how Active Directory keeps itself efficient.

As you will see, a distinct path is cut through your network. DCs in Active Directory do not actually replicate with every other DC. Instead, we have the concept of **replication partners**. (We'll define some terms in a moment.) If you work on a large network, you might have more than 100 DCs, and you might be thinking that it will take a lot of work to maintain those partners. What happens if you have a system crash in the middle of the day—does replication stop working? Actually, no; replication can heal itself in these circumstances. What's more, it's a totally hands-off process. Understanding this process will help you decide whether you should override what the system is trying to do.

A Windows 2000 network has two parts (actually, all BackOffice applications also have two parts, the logical model and the physical model). The first part is the logical model, which is a representation of how you want things to work, without regard to the physical nature of your network. For instance, you want to make sure you are able to support

your administrative structure—and you might make decisions based on the system administrators' requirements. On the other hand, you ignore the physical aspects of your network at your peril. You simply cannot ignore the fact that your connection to a branch office is a T1 (1.54MB per second) link. You must take into account the amount of bandwidth available at any given point in the day. Active Directory wants to move data around your network at any time, so it is up to you to let it know where those slow links are, and how often they can be utilized.

The good news is that we address all of these questions, as well as many others, in this chapter. Active Directory provides an astonishing amount of automation, and you can also tweak it to your heart's content. In this chapter, we are going to explain this process and show you the tools you can use to look at what is really going on under the covers.

Learning Active Directory replication can be a daunting task. You must learn a lot of new terminology, and getting a handle on the precise process is difficult if you are totally new to it. In order to help you fully *see* what's going on, we have decided to show you the entire process in graphic form as figures. Of course, at first glance you might find it a bit overwhelming; however, in trying to decide how best to describe what is undoubtedly a complex process, a picture often speaks 1,000 words. Therefore, not only do we fully describe what is going on each step of the way, but the figures in this chapter provide a visual representation of what we describe. They start fairly simply, and then become more complex as they go on.

Before you can understand the process, you'll need to learn some new terminology. In fact, if you are learning about Active Directory replication for the first time, it is essential that you read this entire chapter and make sure that you understand the terminologies described in the next section. If, however, you are already familiar with the terminology and with what each part does, and you simply need a refresher on Active Directory replication, you might want to skip forward to the figures that explain the process.

Elements of Active Directory Replication

Once you have a clear understanding of the following terms, you will be ready to jump in and see the replication process. Believe it or not, the important terms described in this section cover just about everything you will need to know about Active Directory replication—so let's make sure we have good definitions for them.

Below is a list of terms we will use in our discussion of replication. Following the list is a detailed description of each term. The terms are:

- Domain controller (DC)
- Originating update
- Replicated update
- Update Sequence Number (USN)
- Replication partners

- High-watermark table
- Up-to-date vector table
- Property Version Numbers (PVN)
- Propagation dampening
- Knowledge Consistency Checker (KCC)
- Site links
- Convergence/latency

Domain Controller

Some of this information was covered briefly at the beginning of this chapter. It is repeated here for good reason, because the concept of DCs has changed quite drastically. Microsoft Windows NT also has a concept of domain controllers. In earlier versions, there are two types: Primary Domain Controllers (PDCs) and Backup Domain Controllers (BDCs). From a user perspective, these two types do basically the same thing—they authenticate you when you log on to the network. That is, you are first prompted for a user name and password, and then the name and password you type are checked against a database to make sure you really are who you say you are.

From the system administrator's point of view, the roles of these two types of DCs are very different. PDCs have a read/write version of the security database on them, while BDCs just have a read-only version. This means that, whenever administrators want to change a password or add a new user account, they have to connect to the PDC—no matter where it is on the network. Of course, this process is "invisible" in that administrators never actually have to choose the PDC specifically—but under the covers, that is what's going on.

This process causes some scalability issues. Think about this situation: The PDC for a domain is in the head office in Houston. You are a system administrator sitting in Northern Virginia making the change. Your computer has to connect to the PDC 3,000 miles away! Although the wonders of a network mean this connection can happen in a matter of seconds, it also causes network traffic to traverse many different pieces of cable and to cross probably several routers. It works, but it is not ideal.

With Windows 2000, Microsoft has addressed this problem by adopting what is called a **multi-master** approach. In this case, all DCs hold a read/write copy of the database. Likening this setup to Microsoft Windows NT, one would have to say that each DC in a Windows 2000 network is a PDC.

This approach solves the problem of scalability that applies to earlier versions. It does so by allowing administrators to make changes—such as user name changes—at a DC that is local to them. The administrator no longer has to send data across long distances to do simple administrative tasks. Of course, the approach also introduces another problem: If changes can happen anywhere on a network, then how do all the other DCs find out

about those changes so they are up to date? That's the subject of this chapter—you'll understand the process after you get through all the information here.

Originating Update

We have already said that, in a Windows 2000 network, changes can occur at any DC. That is not to say that all changes are created equally. Two types of changes can occur, and the effects of each type of change are recorded in different places. The first type of change is an **originating update**. Put simply, an originating update is the first time a change is made to a property in Active Directory.

Think about this example: You have two DCs, and you log on as administrator and make a change to a user's password. That change must be written to the database. This is the first time that Active Directory knows of the change. At this point, no other DC is aware of what you are doing. Therefore, this is the originating update.

Replicated Update

Following from the previous definition, we have what is known as a **replicated update**, the second type of change. A replicated update is a change made to Active Directory that did not originate at that copy. Once a change has been successfully written to the first DC, the change must be sent to other DCs. When that change is sent, it will be written to the respective copies of Active Directory. Because the change did not originate at the DC, the change is considered to be a replicated update.

As you'll see later in this chapter, this subtle difference can cause other mechanisms to kick in. In some ways, DCs respond in the same manner to both originating updates and replicated updates. On the other hand, this difference can cause unique events.

Don't be misled by the terms **originating update** and **replicated update**. They are merely efforts to explain that a change occurs somewhere in Active Directory for the first time, and is then copied everywhere else. It need not—necessarily—be what is traditionally thought of as a write. For instance, the deletion of an object is technically also a write, because the change causes new data (in this case, a deletion) to be written to the Directory. If you are connected to a DC and you delete an object, that action would be considered an originating update, in the context the term is used here.

Update Sequence Number

Every domain controller has to keep track of the number of changes it has made to its own copy of Active Directory. An Update Sequence Number (USN) is a 64-bit number that keeps track of changes as they are written to copies of the Active Directory. As changes are made, this number increments by one.

The number increments for several reasons, not least of all because doing so gives the DC a mechanism to let other DCs know it has made a change. The USN is one of the

fundamental building blocks of Active Directory replication. This number *is unique to each DC*. This means that it is not shared among DCs. As time goes on, you will inevitably see that each DC has a different USN. You might have one DC with a USN of 400, and another with a USN of 5. This is not a problem—remember, the USN starts at 1 and increments by one for every change made. If you have a DC with a USN of 400, then this DC may have been online for a long time. A DC with a USN of 5 might be new. USN values can also change if a DC has been restored from backup.

It is not important that all DCs have the same number of changes, only that the USN increments by one for each change. Make sure that you understand this concept—if you don't, then you have missed a key element used in replication.

The USN increments for *every* change made at a DC, including both originating updates and replicated updates. It does not matter where the change came from—this number will increment by one for each occurrence. You might wonder what happens when this number grows to its maximum size—after all, a 64-bit number is finite. You needn't worry: A 64-bit number means that you could write 10,000 changes per second and not run out of numbers for 66 million years. Anyone want to take a bet that there will be a new version of Windows 2000 on the market before then?

Replication Partners

Every DC can accept a change, and it then replicates the change around your network. Sounds simple, doesn't it? But, how is a server in London going to contact and replicate with a server in Houston? It probably won't. A **replication partner** defines a direct replication relationship. That is, if a DC sends data directly to another DC, the two DCs are said to be replication partners.

It is a key point to remember that in Windows 2000, every DC does not talk to every other DC. Instead, you end up with many little hops that make sure data is propagated everywhere. A single DC might replicate directly with one or several different partners. Those partners, in turn, will have different partners that they talk to, and so on. In the end, data is replicated throughout the network.

If you think about your network for a moment, you will see that this process makes perfect sense. First, unless you work on a very small network with few DCs in a single location, all of your DCs will not have a fast connection to every other DC. Consider remote offices, or DCs in foreign countries. It really does not make sense for a single DC to talk to every other one. Best let Active Directory build its own replication topology, as we'll discuss in a moment.

High-Watermark Table

Now you know that DCs band together into units called *replication partners*. But how does one controller know that a change has been made to a partner, and that the change

needs to be replicated? It does this by using two tables that are stored in memory at each DC. One of these tables is the **high-watermark table**.

A high-watermark table stores the name of each of the DC's replication partners, along with the last known USN value for that DC.

The statement we've just made is not quite accurate. Instead of the server name, the value in the high-watermark table is actually the Globally Unique Identifier (GUID) of the copy of Active Directory stored on a DC. Because GUIDs are 128-bit numbers that look like this:

```
a762f268-0dgg-11e4-b5f6-00b0c61e0543
```

we decided to simplify their representation. Although a GUID is an interesting piece of information, it does complicate things when writing about Active Directory replication. Just keep in mind that the actual server name is not stored in these tables.

The high-watermark table is used to make sure that only the necessary number of changes are sent during replication. (This process will be illustrated later.) The important thing to remember is that this table stores a record for *each replication partner only*.

It is worth noting that Active Directory replication is *pull* only. Data is never pushed out. That is, a DC always asks for data to be replicated—the data is never forced upon the DC.

Up-to-Date Vector Table

The up-to-date vector table lists every DC on the network, along with the USN of the last originating update made on that DC. Let's analyze that statement a little bit.

We just looked at the high-watermark table, and it seems to contain similar information. But actually it doesn't—and this area can quickly become confusing to those who are new to Active Directory replication. Let's contrast these two tables and see where they differ.

In the case of the high-watermark table, the type of write (originating or replicated) that causes the USN to increment is not an issue. It does not matter. A change occurs, the USN increments, and it is now different—period. Also, entries are made only for those DCs that are replication partners, not for every DC in the domain. That is the extent of the USN when applied to the high-watermark table.

In the case of the up-to-date vector table, the type of write that occurs *is* important. This table contains an entry for every DC in the domain, along with the USN at the time of the last originating update. (If you don't remember the difference between an originating update and a replicated update, then refer back to our definition earlier in this chapter.) Essentially, the up-to-date vector table keeps a record of the last time a change was first made at each DC.

The purpose for these tables will quickly become clear later, when we present an example that illustrates Active Directory replication.

Property Version Numbers

Much of what you have seen so far consists of efforts to offer different levels of uniqueness. For instance, suppose you have recorded that the last change at a server was the one-hundredth change that occurred there. If the server indicates that it now has had 102 changes, it is obvious that two more have occurred since the last time you heard from the server. In order to work properly, Active Directory needs more information than the simple fact that there was a change. It is not enough that you know *where* something changed, or how many times a server has accepted a change—you also need to know how many times a *specific property* has changed. The reasons will become clear later, as we drill down to the replication process.

The Property Version Number (PVN) is a value that is appended to every property within Active Directory. It traces the number of times a specific property has changed. Properties start with a version of 1. If a change is made to the property, then it increments to version 2. This number is used to make sure that two replicated changes arriving at a DC have a method of working out precedence. A higher version number always "beats" a lower version number.

There are always exceptions to the rule. In the event of a restore operation, it is possible to override this default behavior. However, ordinarily the PVN is a significant tool that you should not override. See Chapter 9 for details on restoring Active Directory and overriding PVNs.

Propagation Dampening

Propagation dampening is a term used to describe the process that Active Directory uses to ensure that changes don't endlessly loop around a Windows 2000 network. The basic principle of replication is a change to Active Directory. As you have seen, such a change causes the USN to increment, which in turn causes a DC to signal its replication partners. Any change—replicated or originating—causes the USN to increment. So, even the process of receiving a replicated update will cause this trigger to fire. Obviously this process is not ideal, because it could trigger a storm of replications on your network, flooding it with traffic.

To understand fully how Active Directory performs propagation dampening, you need to have a good understanding of all the other terms defined in this section. Furthermore, to explain it in paragraphs of text might make for some dry reading. We'll cover this topic in detail later in the chapter.

Knowledge Consistency Checker

By now you should be aware that replication within Active Directory is a controlled process. Instead of broadcasting changes into the network for everyone to hear, DCs

replicate their changes with replication partners. Administering all those links would be a lot of work—but, fortunately, the process is largely automatic.

The Knowledge Consistency Checker (KCC) is a background service that creates links between your DCs. These links are used for replication. It is possible (but not recommended) to override the KCC, but doing so complicates your Windows 2000 network design. What's more, the KCC periodically goes back and reevaluates its replication topology. If changes have occurred on a network (perhaps a new DC has been brought online, or an old one has crashed), then the KCC will recalculate an optimal path and reconfigure Windows 2000 automatically. It is highly recommended that you allow the KCC to build your links.

You might well imagine that this is one of those smoke-and-mirror items you hear about sometimes—you know, like when a vendor tells you that something is automatic, but you really have to do most of the work yourself. Well, for once the hyperbole is accurate. The KCC really does work in the background, making sure that replication is efficient on your network.

The KCC service runs on DCs. However, you won't see an entry for it on the process list, because the KCC runs in the context of the Local Security Authority (LSA). The KCC links together DCs using a proprietary algorithm. This algorithm creates a two-way ring topology that offers fault tolerance to the replication process. Furthermore, the KCC ensures that the DCs it connects are never more than three hops away from each other.

Site Links

A site link is a connector that can be configured between two Active Directory sites. These links have values assigned to them that include a **cost**. This cost is simply a number: The larger the number, the higher the cost. The KCC uses this information when calculating the route that replication should take. For instance, suppose two routes exist from DC A to DC B, and the former has a cost of 2, whereas the latter has a cost of 6. The KCC will favor the first route because it is "cheaper."

Convergence/Latency

Convergence is term used to describe a network in which all DCs are 100 percent up to date: There are no more changes to be replicated. In a Windows 2000 world, this state is something akin to Nirvana. Don't be worried if you never quite reach it (or reach it and can't maintain it for very long). The goal is not necessarily to have no replication taking place, but to ensure that replication is working efficiently.

Latency, on the other hand, describes the inherent delay in replication. When a change is replicated, a communication line must be set up between DCs, packets of data must be sent and analyzed, and changes must be sent and written to Active Directory. All this takes time—albeit a relatively small amount of time. This delay is called *latency*. Latency

is not always a problem, and it occurs all the time. In fact, sometimes you'll cause latency deliberately—for example, when you are connecting different sites (more on this later).

On the other hand, you do not want too much latency. Striking a balance between too much and too little is the job of the system designers, who should pay close attention to minimizing latency (but not to eliminating it).

We have used many new terms in this chapter. Don't worry if you can't memorize them straight away. When you begin to see how they are used, they will make more sense. Although the list is not exhaustive, you now have been introduced to all the concepts you need to understand how replication works.

The Replication Process Illustrated

The terms defined so far in this chapter are the key to your understanding of how replication works within Windows 2000. However, we do realize that reading a list of terms and their definitions is not the optimal way to learn something. You need to know about the *process* of replication. In the following section, you will see replication working in a series of figures.

What follows is a scenario at the InsideIS network. We have a fairly small network at our business, but it includes all the elements you need to learn about Active Directory replication. Let's begin by taking a look at our network. In Figure 14-1, you see our simple network: four DCs named after our four favorite places (Basildon, Dallas, Houston, and London). Each DC is on the same subnet, and therefore, by definition, they are on the same site. Each has fast connectivity to every other DC.

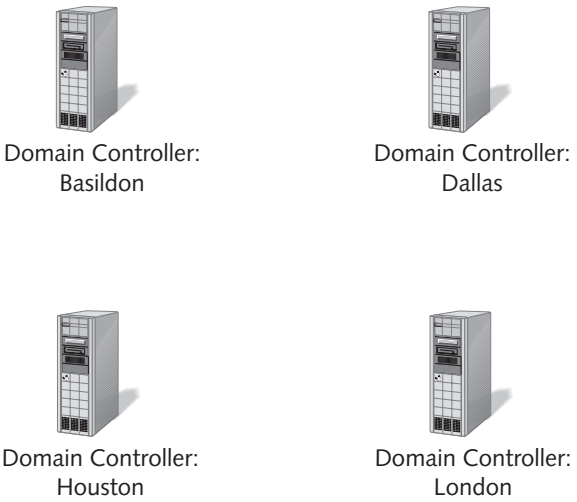

Figure 14-1 Inside IS domain with four DCs

In Windows 2000, every DC in a domain has replication partners with which it shares information. These partnerships form a web that eventually allows every DC to learn about every change that occurs. Let's draw in the replication partners for each of these DCs, as shown in Figure 14-2.

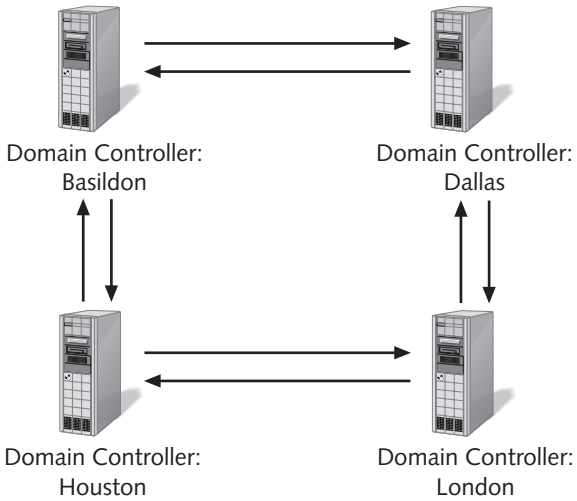

Figure 14-2 Domain controllers and replication partners

In order to help you fully understand what this figure is showing, Table 14-1 details what is shown there.

Table 14-1 Domain controllers and their replication partners

Domain Controller	Replication Partners
Basildon	Dallas and Houston
Dallas	Basildon and London
Houston	Basildon and London
London	Dallas and Houston

Earlier in this chapter we talked about *convergence* and *latency*. In a fully converged network, every change that has been made in Active Directory has been successfully replicated to every DC. Latency is the amount of delay before this replication takes place. In the example, we have a fully converged network: That is, everything is perfect (for the moment).

Figure 14-3 fills out our network diagram to show all the elements we will need for the rest of this chapter. First, we added the USNs to each of the servers. Remember that the USN increments by one for each change the DC accepts. At first glance, you might think that all DCs would have the same number—after all, each DC ends up knowing about every change, so it stands to reason that each DC would eventually have the same USN.

However, many things can cause DCs to have different USNs. For the moment, we are going to assume that each DC has been up for a different amount of time. London was brought online first, and Basildon was brought online last. Therefore, London has had to make more changes than Basildon, and its USN has incremented a greater number of times.

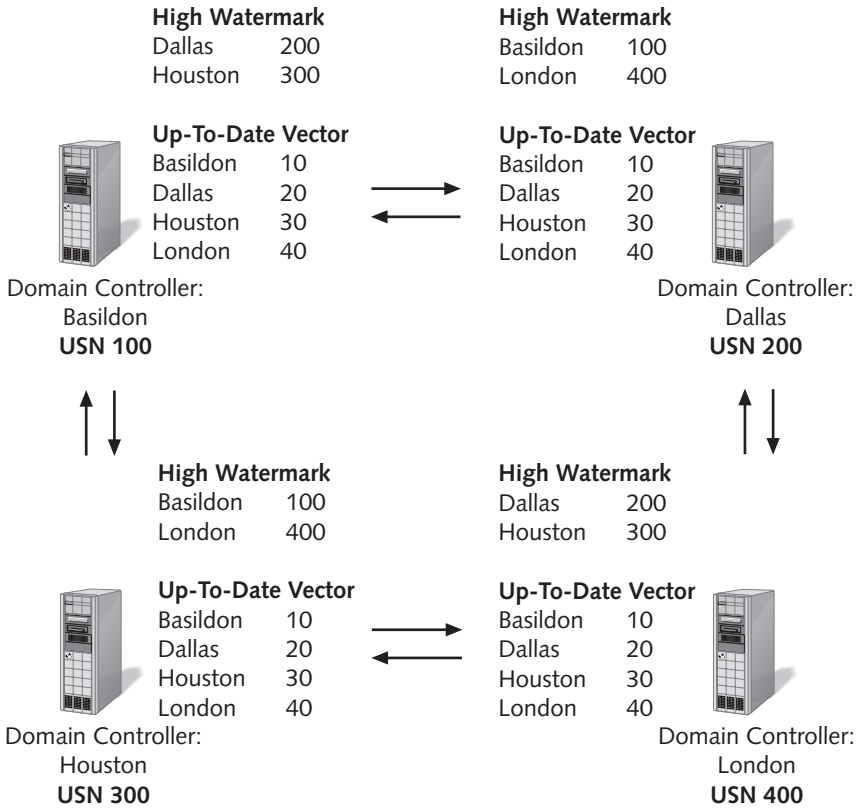

Figure 14-3 A fully converged Windows 2000 domain

We also added the high-watermark table and the up-to-date vector table to each DC. As defined earlier in this chapter, the high-watermark table lists the USNs for each replication partner. Figure 14-2 shows that each DC has two partners, so there are two entries in each DC's high-watermark table.

The up-to-date vector table, you'll recall, contains an entry for every DC in the domain, along with the USN value of the last time the DC performed an originating write. So, there are four entries for each DC. Because this table is concerned only with originating writes, the USN number in this table does not match the actual USN number of a DC.

With this information in hand, we are ready to complicate things a little. We have two administrators we'll call righthand and lefthand, and it should come as no surprise to learn that lefthand doesn't know what righthand is doing—and vice versa.

In our next step, shown in Figure 14-4, lefthand opens the Windows 2000 administration tools and makes a change to the **Description** property for Siobhan, who is a user on our network. Lefthand doesn't care which DC processed the change, and in our case the property change is first made on the London DC.

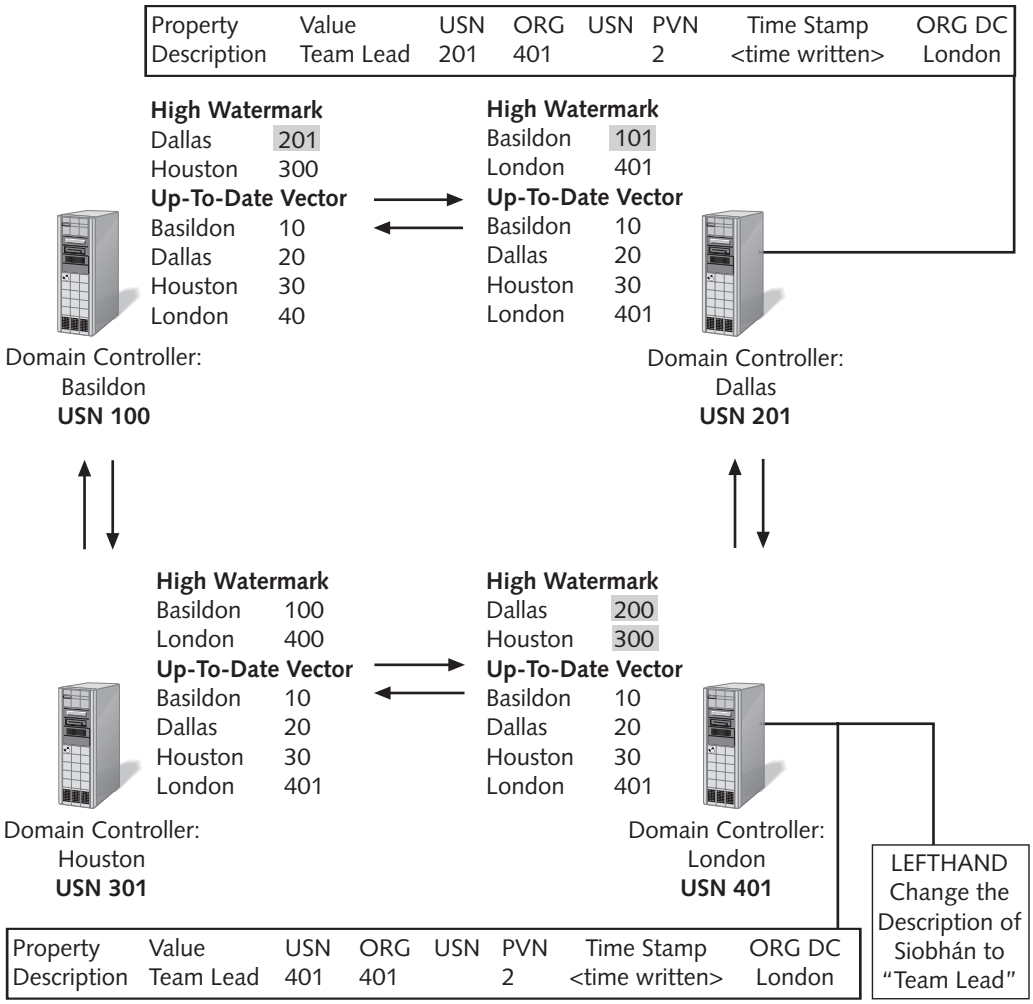

Figure 14-4 Changing a user property

This simple change has actually caused quite a few changes. Let's take a look at what we have added to our figure. Our administrator lefthand changed a property in the DC called London. Because a change is being made on that server, the USN of that server increments by one: The USN for London is now 401. But there is more to it than that—

this is an originating write, so London also writes its current USN in its up-to-date vector table. London's entry in that table now reads

```
London 401
```

This is a common area of confusion: Why didn't that up-to-date vector simply increment by one? The up-to-date vector table simply stores the USN value of a DC *at the time of an originating update*. The table does not attempt to track the number of originating updates that have occurred at the DC—only the USN at the time they occurred.

Figure 14-4 also shows the data that has changed. (This data is somewhat simplified for the purposes of the book—for instance, the ORG DC field should really be the DC GUID. However, that would confuse the issue, so we changed the value to the server's name.)

Note that the PVN of the **Description** property is 2. When the property was created, the PVN was 1; but now the property has changed, so this is version 2. You will see how this comes into play in a moment.

Because there has been a change at the London DC, this DC must send a message to each of its replication partners saying, "I have a change." They will then pull that change from London. Figure 14-5 shows what the picture looks like when this change replicates to both Dallas and Houston.

A lot just happened. The changes that occurred at the Dallas and Houston DCs are similar, so we will use Dallas as our example. First, the change was written to the DC, which in Dallas caused the USN number to increment to 201. Because the change originated in London, the up-to-date vector for London was also updated. This change was not an originating write at Dallas: It was a replicated write. Therefore, the up-to-date vector for Dallas did not change.

The high watermark for London also changed, because the high watermark records the last USN number that Dallas knew about from London. Because this number is now different, an update occurred to reflect the new value.

You should note the data that was replicated to Dallas. The USN number of the write has changed to reflect the USN number at Dallas when this change was made. However, the ORG USN number remained unchanged. Also, the PVN did not change.

The values in London's high-watermark table also changed. They now reflect the new USNs of both its replication partners.

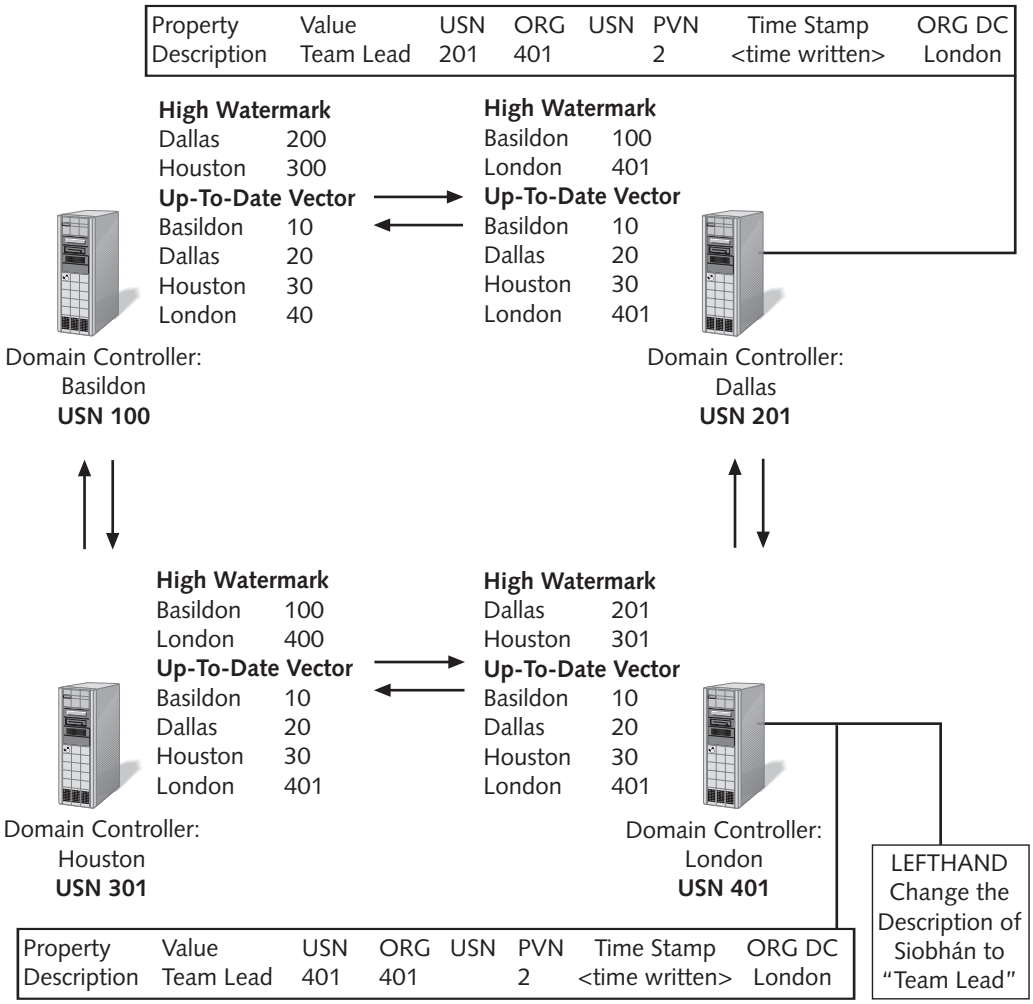

Figure 14-5 Change is replicated to Dallas and Houston

Now, things are going to get interesting. The change is safely made at Dallas and Houston, but Basildon has not seen it yet. Basildon is a replication partner for both Dallas and Houston, and it can get the change from either; but, of course, one of them will be first. In this case, we will assume that Dallas informs Basildon of the change, and that Basildon then pulls it from Dallas. When this occurs, we will have the circumstance shown in Figure 14-6.

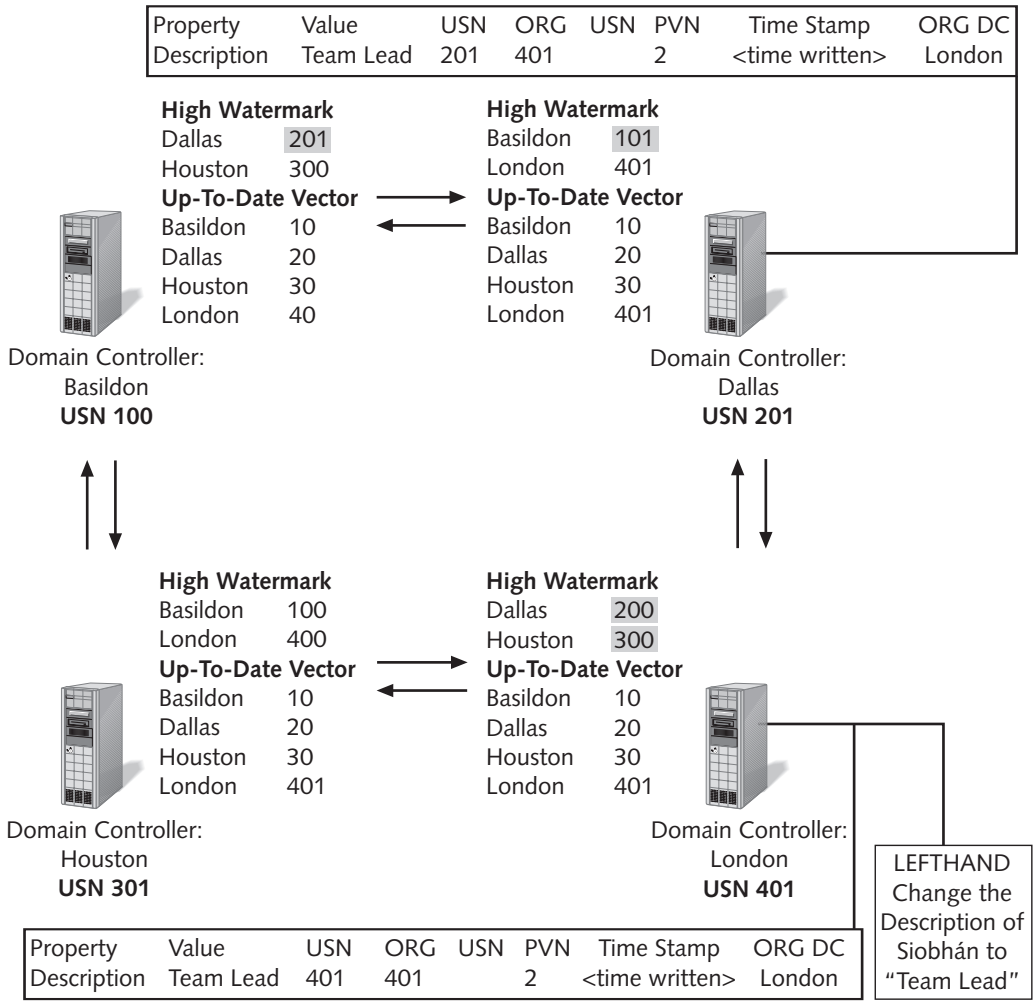

Figure 14-6 Change is replicated to Basildon

Let's examine what occurred at Basildon. The USN for Basildon incremented by one to 101. Replication also caused changes to both the high-watermark and up-to-date vector tables. To help you keep track of these changes, let's make a quick list of everything that changed in Basildon:

- USN of Basildon
- High-watermark table entry for Dallas
- Up-to-date vector for London
- High-watermark entry for Basildon stored at Dallas

The change came from Dallas, but Houston wants to replicate to Basildon, too. What's more, it wants to replicate the very same information. Let's zero in on that conversation, as shown in Figure 14-7.

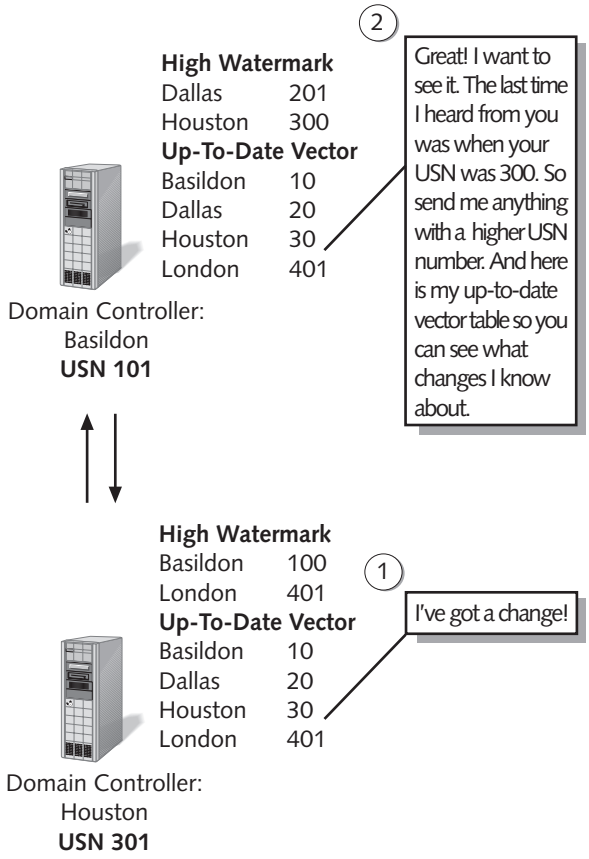

Figure 14-7 The replication conversation

Basildon had quite a bit to say, but where did it get all that information? First, it looked into its high-watermark table and noted that the last time it spoke with Houston, the USN for Houston was 300. So, Basildon asked for changes over 300. It then packaged its up-to-date vector table and sent the table to Houston. The up-to-date vector stores a list of all the originating writes that Basildon knows about. Houston will need this to enforce propagation dampening. Let's see how that works by taking a look at what Houston does when it gets this data from Basildon, as shown in Figure 14-8.

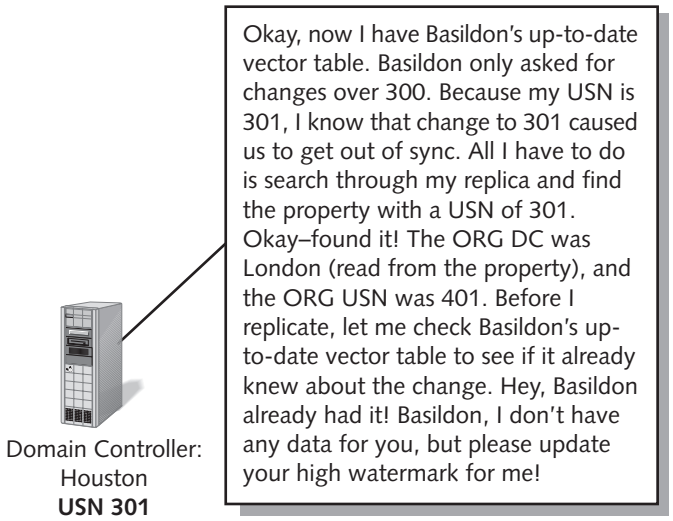

Figure 14-8 Propagation dampening at Houston

It may look like a long conversation is going on at Houston, but the upshot is that redundant data was not sent to Basildon. You can see our domain in Figure 14-9. Note that the high-watermark table at Basildon has been updated to reflect this conversation.

Now for the tricky part. When Dallas replicated the data to Basildon, the USN number of Basildon incremented. As a result, Basildon will enter the change process itself: It will tell Houston, "I have a change," and Houston will ask Basildon for that change. This process was shown in Figures 14-7 and 14-8. Of course, no data is exchanged, but you should know that the process is triggered.

Once you have been through the replication process a few times, it will become second nature. Of course, we have not covered every eventuality. Let's throw something a bit more complex into the mix.

Replication Conflicts Illustrated

So far in this discussion, we have gone from a fully converged network to a network in which a change was being made. We ended up with a fully converged network again. Now we will have each of our administrators make a change, and see what happens.

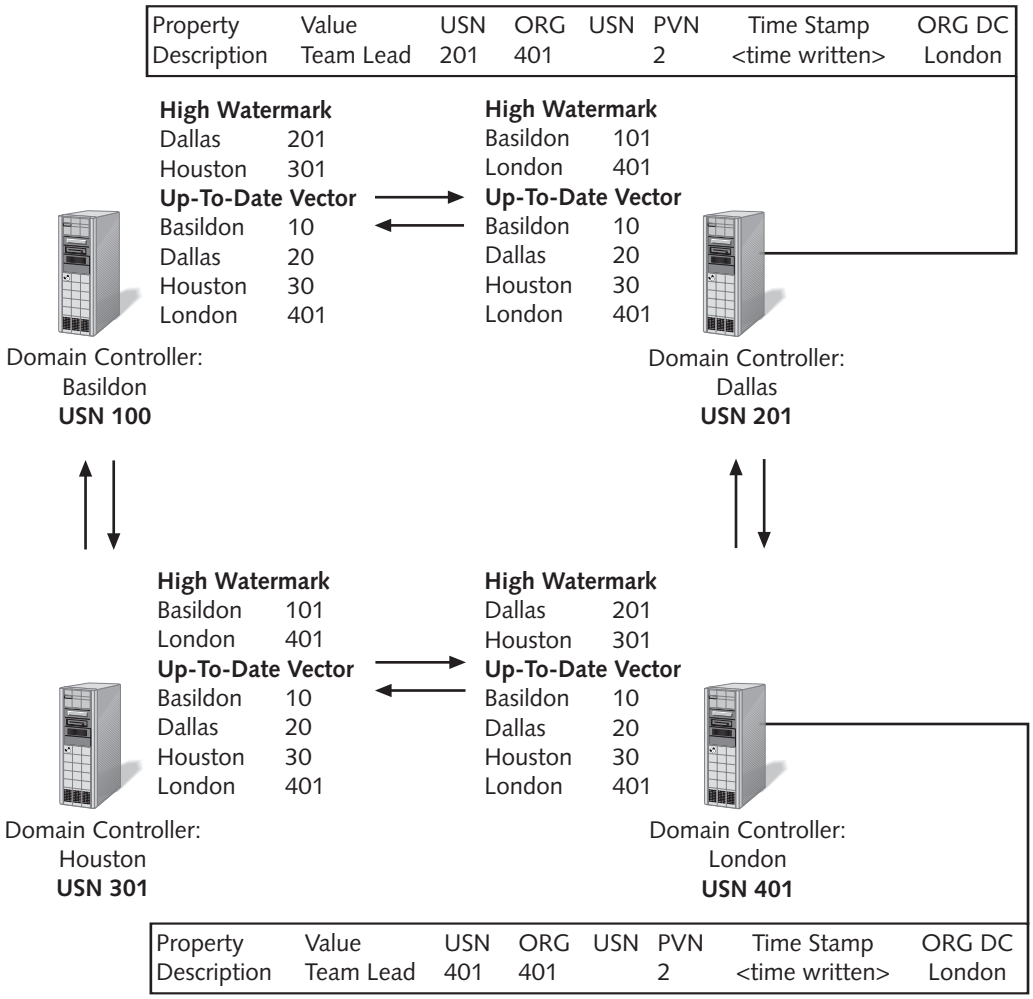

Figure 14-9 A fully converged InsideIS network

In the following example, the administrator in London—lefthand—will update the **Description** property of a user object called Mike. At the same time, the administrator in Basildon—righthand—will make a change to the same property. In Figure 14-10, you can see how things look as the changes are made.

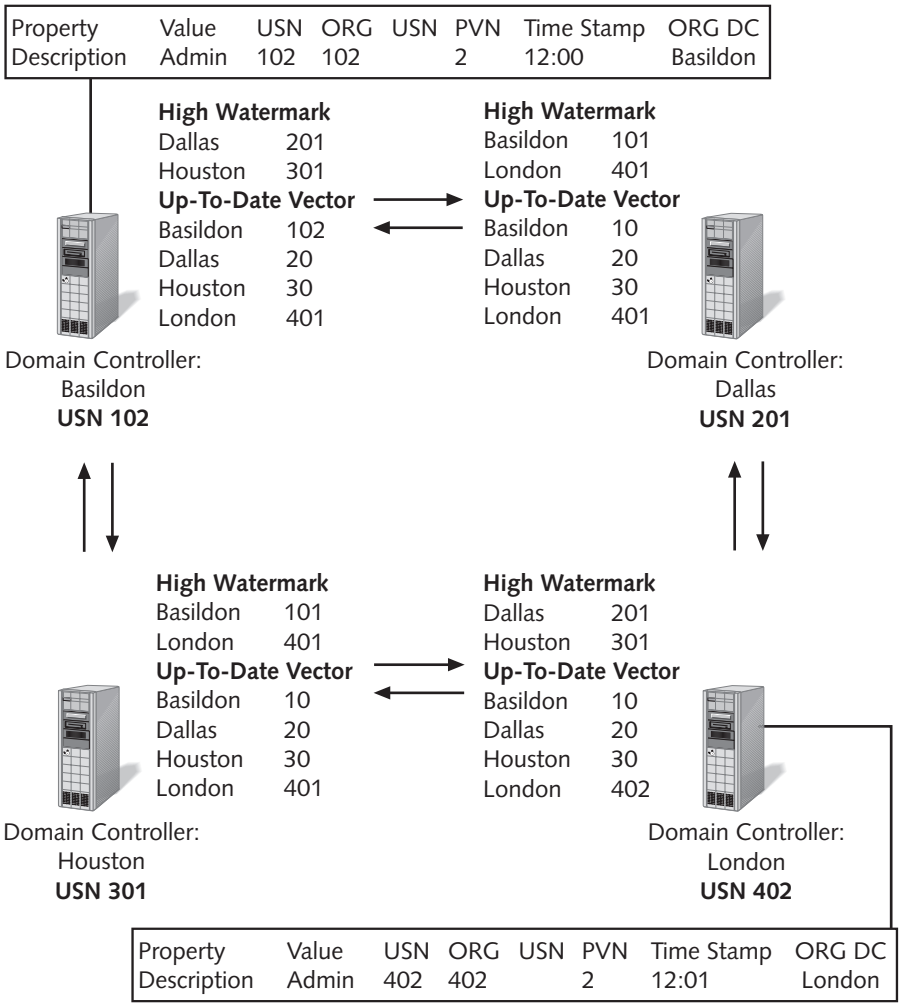

Figure 14-10 Two changes made on the same property

As you have seen, DCs replicate with their replication partners. A partner can (and probably will) have more than one replication partner. Changes that arrive at a DC are dealt with one at a time. In our example, Basildon and London have the same replication partners. For the purposes of this example, we will use Houston to illustrate how replication conflicts are resolved.

A replication conflict occurs when a change is made to a single property at about the same time on two different DCs. In our case, we are back with our trusty **Description** property. Because we've already discussed some of this process, let's jump ahead. In Figure 14-11, you can see that Basildon has successfully replicated the change to Houston (beating London to the punch).

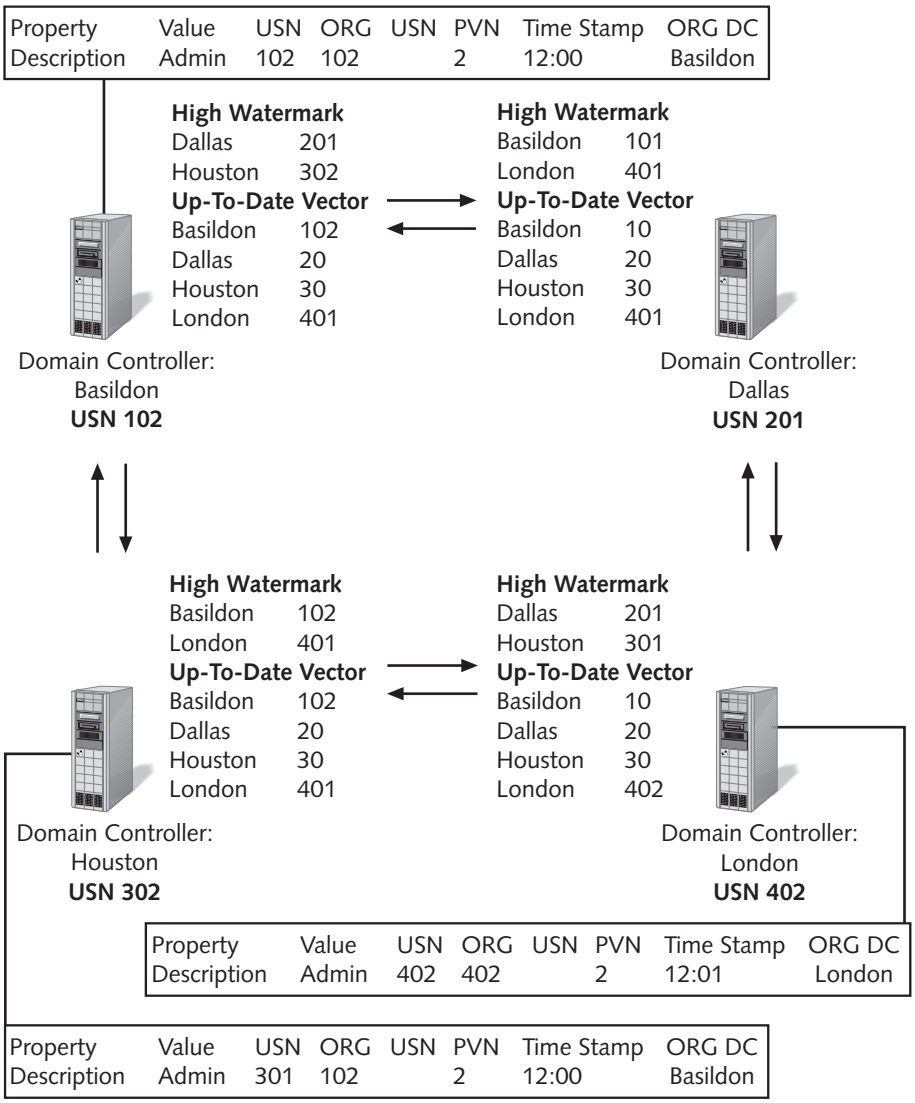

Figure 14-11 The change arrives at Houston

The following items have changed since Figure 14-10:

- The USN of Houston incremented by one
- The high-watermark table changed at Houston
- The high-watermark table changed at Basildon
- The up-to-date vector table changed at Houston
- The property changed at Houston

Now, let's cause a conflict. Houston has the change from Basildon and now accepts the change from London. First, it looks at the high watermark for London and asks for anything with a USN higher than 401. It also sends its up-to-date vector table so London can perform propagation dampening.

In this case, as far as London knows, propagation dampening need not take place. London doesn't yet know about the change made at Basildon, so it is going to replicate its own data. Once this analysis has taken place, the updated property will be replicated. The rest of this work occurs at Houston.

Figure 14-12 (shown on page 385) shows what our situation looks like once the data has been sent from London to Houston.

Houston now has another change to the same data. To figure out whether it should make this change in Active Directory, it uses a three-step process. If the first step fails, it moves on to the next step. These three steps are as follows:

1. Compare PVNs
2. Compare date and time information
3. Compare Active Directory GUIDs

Let's apply this process to our example so we can see how it works. First, Houston compares the PVN of the information it just received from London with the PVN of the property it just wrote. If the London PVN is higher than the value from Basildon, then the game is over: The London data wins. In this case, the PVNs are the same.

Next, Houston compares the date and time of the two sets of information. If they are the same (within one second of each other), then Houston will move on to the third and final parameter. In our case, however, the change in London happened one minute after the one in Basildon. Therefore, the London data overwrites the change made at Basildon.

If necessary, the final step compares the GUIDs of each Active Directory replica. Whichever is higher will win. It is unlikely that this would ever be the determining factor, however, because there is little chance of changes being made within one second of each other.

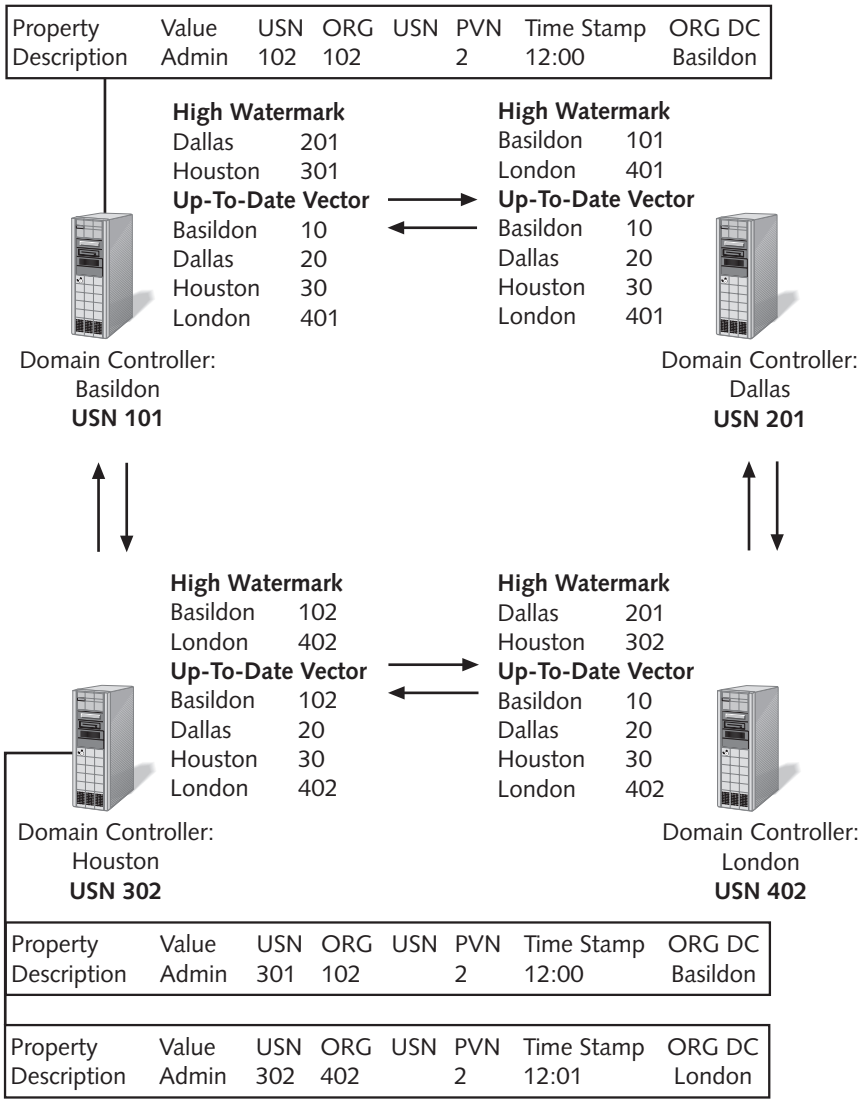

Figure 14-12 Data replicated from London to Houston

Just for the sake of completeness, Figure 14-13 (shown on page 386) illustrates what our figure will look like once everything is nicely converged again. When convergence is achieved, all changes are written to all DCs.

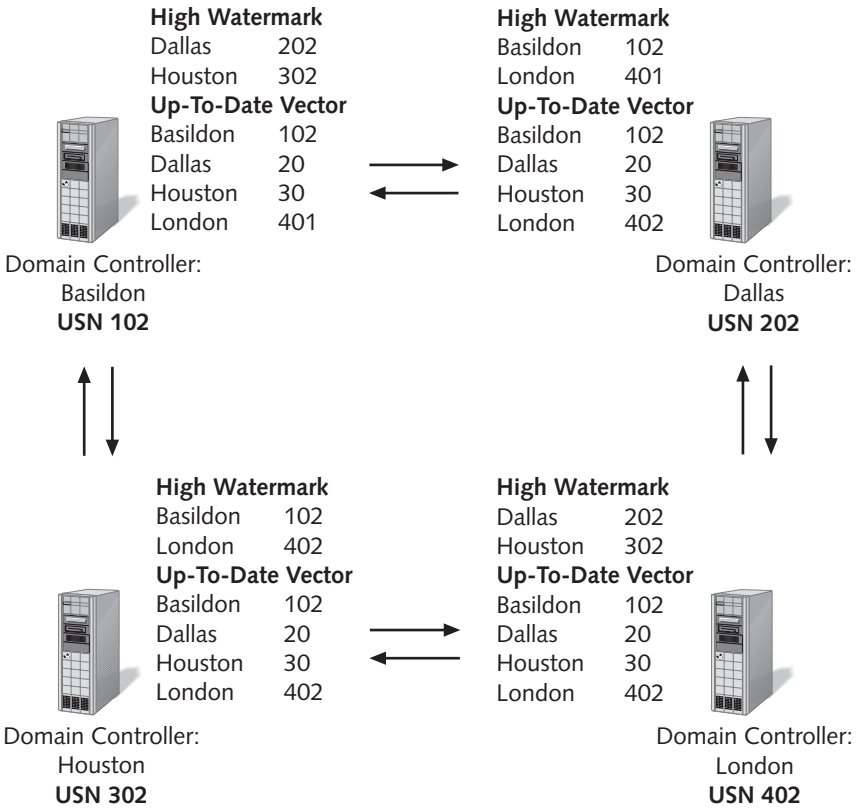

Figure 14-13 Fully coverged domain

Intersite and Intrasite Replication

We need to discuss one final topic. In order to simplify what is, in all honesty, a somewhat complex task, we used a fairly simple example earlier in this chapter. It included a few DCs that each had fast connectivity (10Mbps or more) between them. Also, our DCs were in the same domain and in the same subnet, and therefore they could replicate whenever they wanted. Of course, life is never that easy, so we need to fill in the blanks.

For a better real-world view of replication, we need to take into consideration the effects of slow links (links slower than 10Mbps) on replication. Obviously, replication across these links cannot happen with the same frequency as it can across fast links. Windows 2000 helps by defining both physical aspects of your network (by IP subnet) and then allowing you to apply costs and a schedule that dictate when replication can occur. IP subnets are grouped into sites.

Windows 2000 sites were discussed in Chapter 2. A **site** is a group of well-connected DCs. This group is defined by IP subnets. Active Directory replication works in slightly

different ways depending upon the location of the DCs to which the data needs to be replicated.

Replication within a site (**intrasite replication**) happens as quickly as possible. The focus of Microsoft's design is to get the data replicated quickly. Because the DCs are well connected, bandwidth is not an issue. For **intersite replication** (replication between DCs in different sites), replication is geared toward preserving bandwidth. Intersite replication has the following qualities:

- Data is compressed before it is sent.
- Replication occurs on a schedule.

Replication between sites occurs through **bridgehead servers** (also defined in Chapter 2). Basically, there is a connection point between the two sites—this connection point is between a server in one site and a server in the other. Each server acts as a **bridgehead** (hence the name "bridgehead servers") within its site. Active Directory data is replicated between the two bridgeheads. Once the bridgehead server knows about the changes, replication within each respective site occurs normally (as discussed in our example in this chapter).

Chapter Summary

- In this chapter, we took a detailed look at Active Directory replication. We started with a simple overview of the Windows 2000 components that participate in the process. These components are:
  - Domain controller (DC)
  - Originating update
  - Replicated update
  - Update Sequence Number (USN)
  - Replication partners
  - High-watermark table
  - Up-to-date vector table
  - Property Version Numbers (PVN)
  - Propagation dampening
  - Knowledge Consistency Checker (KCC)
  - Site links
  - Convergence/latency

- Each of these components has a role to play in replication. Once we had defined these terms, we took a detailed look at two replication scenarios. First, we noted that replication traffic is not broadcast across the network. Instead, each DC has what is known as a replication partner (or partners)—and it replicates only with these partners. Replication partners can be chosen manually, but you are better off letting the KCC do this for you. In order for the KCC to work, you must define site links with Active Directory, essentially associating costs with slow links. The KCC will examine these site links and the sites you have defined in Active Directory, and will use a proprietary algorithm to work out the optimal path of communication (or replication partners).
- If a DC goes offline or a new DC is installed on the network, the KCC will notice this event on its next cycle. If necessary, it will redefine replication partners for DCs. In any case, the KCC does this by default every 30 minutes. It is a good idea to let the KCC do its work unhindered.
- We next showed a single change and how it was replicated throughout a network. Starting with a fully converged network, we saw a change made at one DC ripple out and be copied to all DCs in the domain. This process directly affects two tables stored in the memory of every DC: the high-watermark table and the up-to-date vector table. We also observed that the USN of each DC incremented as the change replicated.
- We saw that this form of multi-master replication has some problems. Because a change made at a controller causes the controller to want to replicate, you could end up with storms of changes that would hurt the performance of your network. The process that is in place to stop this problem is called propagation dampening. We examined how it works and what mechanisms are involved in the resolution.
- We looked at the process that takes place when a change is made to the same property at two different DCs, at about the same time. Eventually, these changes will clash at a DC and need to be resolved.
- The three tie-breaker mechanisms are:
  - Comparing PVNs
  - Comparing times and dates
  - Comparing Active Directory GUIDs
- These steps act as a filter. The first comparison is applied; if that fails to resolve the issue, then the second is applied; and so on. Because no GUID can be duplicated, this system guarantees that a conflict can be resolved.
- Finally, we discussed the difference between intrasite replication and intersite replication. Intrasite replication refers to replication that takes place within a single site. Intersite replication occurs across multiple sites. Intersite replication differs from intrasite in that the data being replicated is compressed before being sent (intrasite replication data is not compressed). Also, replication between sites occurs on a fixed schedule.